On-Line Testing and Diagnosis of Microcontroller

Khaled Elshafey, Ahmed Elhosiny
System and Computer Engineering Department
Faculty of Engineering
Al-Azhar University
Nasr-City, Cairo, Egypt
E-mail: kh_elsha1@yahoo.com, aelhossi@uoguelph.ca

***Abstract* - This paper presents an on-line testing and diagnosis approach of microcontroller. The proposed approach has been achieved through both fault masking and fault diagnosis algorithms. Concurrent tesing technique through using triple modular redundancy (TMR) is required to mask the operational faults and specially tolerate the transient faults. For permanenet faults and in parallel with TMR, an on-line and non-concurrent fault detection and diagnostic technique is used to locate the faulty elements. The fault detection and diagnostic technique uses a set of assembly programs that test the entire microcontroller instruction-sets called macros. The macros are able to excite all of the microcontroller functions. A macro is associated to each machine-level instruction; and composed of a few instructions, aimed at activating the target instruction with some operand values representing the macro parameters and propagates the results of its execution to an observable memory positions. A Simulation study has been done using Xilinx Foundation tool, VHDL, and an FPGA Vertix chip.**

***Index Terms*- Diagnosis, FPGA, Microcontroller TMR.**

I. INTRODUCTION

During its lifetime, a digital system is tested and diagnosed on numerous occasions. For the system to perform its intended mission with high availiability, testing and diagnosis must be quick and effective. A sensible way to ensure this is to specify test as one of the system functions- in other words, self-test. Reliability, availability and serviceability (RAS) are the major factors in consideration in system design to provide continuous correct operation [1].
Since faults cannot be completely eliminated, critical systems always employ fault tolerance techniques to guarantee high reliability and availability.

fault tolerance (FT) techniques try to keep the system operational despite the presence of faults [2]. FT can be achieved through hiding the occurence of faults and preventing it from generating errors (fault-masking), or through fault detection and fault repairing.
Since faults cannot be completely eliminated, critical systems always employ fault tolerance techniques to guarantee high reliability and availability.
Technology advances made it possible to integrate on a single chip enormous numbers of transistors, thus allowing the inclusion on a single chip of entire systems, including microprocessors (MP), Microcontroller (MC), memories, ASICs, and peripherals. The development of this new class of systems, called Systems On Chip (SOCs) [3].
Testing is a crucial issue in SOC development and production process. The popular solution for SOCs that include MP and MC cores is based on making them execute a test program [4]. Generating a test sequence for the MP based on functional testing is a challenging task, usually much harder than for a generic ASIC. In fact, the internal structure of a MP and its behavior make unfeasible the adoption of a standard automatic test pattern generation (ATPG). In particular, the internal structure is based on a sequentially complex decode and control unit that decode the instructions and sends the appropriate control signals to a large data-path, which may include hard- to –test elements such as multipliers and dividers [5]. The two main components of a MP require different approaches for their test: the decode and control unit can be seen as a complex finite state machine (FSM), and its test requires suitable sequences to traverse its state graph to excite and observe possible faults. Therefore, its test is mainly a sequential problem. Finally, test sequence generation for MP necessarily requires the knowledge of the processor instruction set and instruction format since only correct programs can internally perform meaningful operations.
A MC is a multipurpose, programmable device that reads binary instructions from a storage device.

1-4244-0765-6/06/$20.00 ©2006 IEEE

Memory, accepts binary data as input and processes data according to those instructions, and provides results as output. MC is every thing a microprocessor has as well as timers, parallel & serial I/O ports, and internal RAM and ROM [6]. MCs are intended to be special-purpose digital computers can be implemented on a single VLSI chip. Each MC has a fixed set of instructions in the form of binary patterns called a machine language.

A field programmable gate arrays (FPGAs) are a chip in which the final logic system structure can be directly configures by the end user. They are a general purpose chips that can be configured a wide variety of applications. FPGAs represent an important part in many design concepts like SOC design, hardware-software co-design, and the adaptive computing systems (ACSs) [7].

This paper presents a design and implementation of an embedded fault tolerance VHDL-microcontroller on FPGA based on redundancy and reconfiguration techniques.

The paper is organized as follows. Section 2 outlines the MC design process. The MC fault model was presented in section3. The embedded FT design and implementation based on fault masking and reconfiguration was introduced in section 4. Section 5 presents the reliability calculations, while section 6 includes the conclusions.

II. MC DESIGN AND IMPLEMENTATION

The proposed design is an eight-bit MC. It consists of 8- bit CPU with 64 Different single byte instructions, 16 x 8 ROM (program control memory) where programs are stored, 8 x 8 RAM (data memory) whose address space contains two 8 I/O ports as addresses to facilitate dealing with I/O, and Multiplexer 2 x 1 to control the fan in data input to CPU.

A. CPU Implementation

The CPU consists of seven main parts: Registers (address, data, instruction, accumulator, status, program counter, four temporary registers), 8 bit Arithmetic and Logic Unit (ALU), Control unit (CU) with 38-bit control signal, Latchs, Multiplexers, Demultiplexer 1x 8 for bit addressing operation , and Comparator. The block diagram of the CPU is shown in Figure 1.

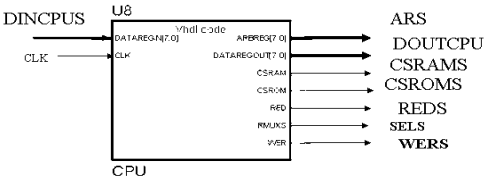

Fig. 1. CPU Block Diagram

B. ROM Implementation

A 16 x 8 Rom is used for holding the machine instructions . The instructions in ROM are permanent as shown in Figure 2.

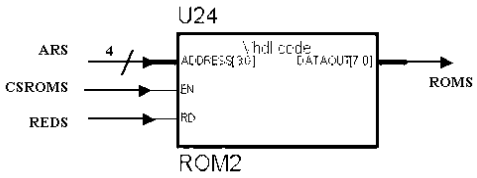

Fig. 2. ROM (16 x 8)

C. RAM Implementation

The RAM is used to hold data that will be processed, and two addresses of its address space are viewed as I/O ports. RAM is the "waiting room" for the computer's processor. RAM holds raw data that is waiting to be process as shown in Figure 3.

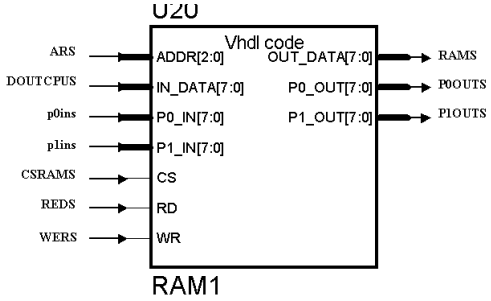

Fig. 3. RAM (8 x 8)

D. MC Implementation Results

A simulation study has been done by executing some instructions stored in ROM. These instructions for instance are as follow:

```
00110000    --00H------MOV[ R A] ,P0     30  H
00000110    --01H----  ADDRESS FOR PO     6H
00100100    --02H----  MOV B DATA        24 H
11100011    --03H---       DATA=        E3   H
```

Where, the value 30H present in port 0 moves to file register A, and DATA=E3H moves to file register B

The implementation using Xilinx Foundation 3.1i on Virtex XCV1000 FPGA chip utilized 292 slices out of 12288 (2%) with maximum path delay of 37.696 ns.

III. FAULT MODEL

The main requirement for the choice of the fault model is that the model should be able to capture the change in functionality caused by most of the commonly occurring physical (operational) defects in the circuit. According to their stability in time, operational faults can be classified as: permanent, intermittent, and transient faults [8].

Faults affecting the operation of a MC can be divided into the following classes [5]:

- Addressing faults affecting the register-decoding function, which leads to no register is accessed or a set of registers (that may or may not include the decoding register(s) is accessed.
- Addressing faults affecting the instruction-decoding and the instruction-sequencing function allowing for partial execution of instructions and for execution of new instructions, not present in the instruction set of the MC
- Faults in the data-storage function as stuck-fault model. It allows any register in the MC to have any number of bits stuck-at 0 or stuck-at-1.
- Faults in the data-transfer function, as if any line in a transfer path can be stuck-at 0 or stuck-at-1, or any two lines in a transfer path may be shorted.
- Faults in the data –manipulation (processing elements) function.

IV. PROPOSED TESTING AND DIAGNOSIS APPROACH

Fault masking technique, for instance TMR was embedded within the most important MC's component like control unit and ALU. The TMR technique detects the operational faults and specially tolerates the transient faults. For permanenet faults, a non-concurrent fault detection and diagnosis technique is used in parallel with TMR to locate the faulty elements. The faulty element(s) located by the Macro procedures is then circumvented by a new configuration downloaded from an external memory [9]. This is possible because some of the FPGA configurable logic blocks (CLBs) are used as spares for the rest. Therefore, the system will never go to the case of voting failure except when the majority modules are failed at once. In case of TMR voting failure, the system ceases and a diagnosis algorithm is used to locate the faulty element(s).

A. Testing via TMR

The concept of redundancy implies the addition of information, resources, or time beyond what is needed for normal system operation [2]. The redundancy can take one of several forms, including hardware, software, information, and time redundancies. The physical replication of hardware is perhaps the most common form of redundancy used in systems. As semiconductor components have become smaller and less expensive, the concept of hardware redundancy has become more common and more practical.The fault masking technique concept is to hide the occurrence of faults and prevent the faults from resulting in errors. The most common form of fault masking is the TMR. The basic concept of TMR is to triplicate the hardware and perform a majority vote to determine the output of the system. If one of the modules becomes faulty, the two remaining fault free modules mask the results of the faulty one when the majority vote is performed. The primary difficulty with TMR is obviously the voter; if the voter fails, the complete system fails. In this article, the voters were considered as faulty free. The TMR technique applied only on ALU and CU as a case study as shown in figure 4.

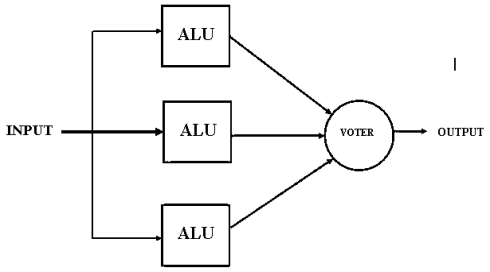

Fig. 4. TMR for ALU

B. MC Implementation and Simulation with TMR

A simulation study has been done by executing some instructions stored in ROM for the three ALU and CU modules. A fault has been injected to one of the three ALU modules and the voter selects one from the majority of ALU and CU modules.
The implementation of MC with TMR on Vertix FPGA chip using Xilinx Foundation tool 3.1i series has been utilized 532 out of 12288 slices (4%) with path delay 36.205 ns. The area overhead is mostly duplicated due to the TMR , while time dealy is closely the same.

C. Diagnosis Algorithm

Diagnosis strategy is depending on a small programs called "Macros"aim at activating the target MC fault model. Each macro activates a certain fault and at verifying the correctness of the instructions execution through propagating the outputs to one or more result words.

Fault Cases:
Generating macros for testing a certain fault cases are shown in the following fault examples:
F1- testing of address file register (AFR) decoding. AFR is consists of four registers A, B, C, and D.

Macro 1:
```
MOV A,[D]       -- INPUT DATA 00H
TARNSFER A
JUMPZ           -- JUMB IF ZERO =1
06 H            -- ADDRESS
INC A
MOV [P1OUT], A -- OUTPUT DATA
```

This macro is for testing the access of register A by firstly loadind 00H into A, transfer operations treat only with A, and finally the result of of the operation in A is tested. The next macro is for testing register C and D based on register A and B.

Macro (2):

```
MOV C,[D]
MOV D,[D]
MOV A,C
MOV B,D
SUB
JUMPZ
INC A
MOV [P1OUT],A
```

F2- Testing of Program Counter (PC) is based on loadind data into register A, repeat increment operation, and then out the data in A. If data in A is increased as the same repeating times, then the PC is faulty free.

Macro (3):

```
MOV A, [D]
INC A
INC A
INC A
.....
MOV [P1OUT],A
```

F3- Testing of an ALU operations as ADD, and XOR are shown in macros (4),(5) respectively.

Macro (4):

```
MOV A, [D]
MOV B, [D]
ADD A,B
JUMP C
INC A
MOV [A],A
```

Where, if final value in register A is changed then the ADD operation is correct.

Macro (5):

```
MOV A, [D]   -- data = FF H
MOV B, [D]   -- data = 00 H
XOR A,B
MOV [A],A    -- output data
```

Where, the XOR logic function is correct if the final result of A is the same as FF H.

V. CONCLUSION

Our testing and diagnosis approach detects and locates the faults on-line through using both of fault masking (e.g., TMR) and instruction macros. The proposed approach had been implemented on an eight-bit microcontroller system as a case study. TMR technique is for masking transient and intermittent faults. In parallel with TMR and non-concurrently with MC system execution, a fault diagnosis technique is used. The fault diagnosis strategy is based on macros. The complexity of the work for developing macros that aim to activating the MC instructions and verifying the correctness of the instructions execution are much lower than for other approaches based on functional testing. The area overhead due to TMR of certain MC elements (ALU and CU) was nearly duplicated, while the maximum path delay was closely equal. However, our approach with using TMR and macros enhances the mean time to failure; therefore it can be used for the long-life mission.

REFERENCES

[1] V. D. Agrawal, C. R. Kime, K. K. Saluja, " A tutorial on Built-in Self-Test", Part 1, IEEE Design & Test of Computers, March 1993, pp 73-82.

[2] D.K. Pradhan, Fault-tolerant computer system design. Prentice Hall PTR Publishing Company, 1996.

[3] P. Magarshack, “Improving SoC Design Quality Through a Reproducible Design Flow”, IEEE Design & Test of Computers, 2002, pp 76-83.

[4] C.A. Papachristou, F. Martin, M. Nourani, “Microprocessor Based Testing for Core-Based System on Chip”, ACM/IEEE Design Automation Conf., 1999, pp. 586-591.

[5] M. Abramovici, M. A. Breuer, A. D. Friedman, “Digital Systems Testing and Testable Design”, IEEE Revised Printing, 1990 by AT&T.

[6] K.J. Ayala, The 8051 Microcontroller Architecture, Programming & Applications. 2nd Edition, Delmar Publishers Inc., 1996.

[7] N.R. Saxena, S. F. Gomez, W. Huang, S. Mitra, S. Yo, E. McCluskey, “Dependable Computing and On-Line Testing in Adaptive and Configurable System”, IEEE Design & Test of Computers, Jan-March 2000.

[8] C. Weaver, T. Austin, "A Fault Tolerant Approach to Microprocessor Design", IEEE Transactions on Computer 2001, pp. 411-420.

[9] K. Elshafey, “Embedding Fault Tolerance via Reconfiguration in Adaptive and Configurable Systems”, IEEE Proc., 15th ICM 2003, Dec. 9-11, Cairo, Egypt, pp. 370-373.